GP/2825
#6
10/30/02

Attorney Docket No. 1691.1002

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Patent Application of:

Peer JOHANNSEN

Application No.: 10/038,870 Group Art Unit: 2825

Filed: January 8, 2002 Examiner:

For: METHOD OF CIRCUIT VERIFICATION IN DIGITAL DESIGN

**SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN APPLICATION IN ACCORDANCE WITH THE REQUIREMENTS OF 37 C.F.R. § 1.55**

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

In accordance with the provisions of 37 C.F.R. § 1.55, the applicant(s) submit(s) herewith a certified copy of the following foreign application:

German Patent Application No(s). 101 00 433.8

Filed: 8 January 2001

It is respectfully requested that the applicant(s) be given the benefit of the foreign filing date(s) as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. § 119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 10/24/02 By: Richard A. Gollhofer
Richard A. Gollhofer
Registration No. 31,106



700 11th Street, N.W., Ste. 500
Washington, D.C. 20001
(202) 434-1500

#6

# BUNDESREPUBLIK DEUTSCHLAND





## Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

| | |
|---|---|
| **Aktenzeichen:** | 101 00 433.8 |
| **Anmeldetag:** | 08. Januar 2001 |
| **Anmelder/Inhaber:** | Siemens Aktiengesellschaft, München/DE |
| **Bezeichnung:** | Ein Verfahren zur Reduktion von Modellgrößen in Hardware-Verifikationsaufgaben unter Verwendung von Signalbreiten-Skalierung |
| **IPC:** | G 06 F 11/277 |

**Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.**

München, den 10. Oktober 2002
**Deutsches Patent- und Markenamt**
**Der Präsident**
Im Auftrag

Hoiß

A 9161
06/00
EDV-L

Erfindungsmeldung

# Ein Verfahren zur Reduktion von Modellgrößen in Hardware-Verifikationsaufgaben unter Verwendung von Signalbreiten-Skalierungen

Peer Johannsen

Siemens AG, ZT-SE-4, 81730 Munich, Germany
☎ (+49) 89 636 53185
Fax (+49) 89 636 42284
Peer.Johannsen@mchp.siemens.de

**22. November 2000**

**Zusammenfassung** Model Checking befaßt sich mit dem Beweis von Eigenschaften eines gegebenen Systems mittels mathematischer Methoden. Beim Einsatz solcher Methoden in der Hardware-Verifikation an realen Schaltungen ist man mit grundlegenden Komplexitätsproblemen konfrontiert. Es ist allgemein bekannt, daß oft allein schon die Größe einer Schaltung Verifikationsaufgaben scheitern läßt oder die Laufzeiten gegenwärtiger Verifikations-Tools in den Bereich von Tagen treibt, oder aber schon vorher das Herauslösen relevanter Teile aus dem System und eine getrennte Verifikation erzwingt. Mit ständig wachsender Größe digitaler Schaltungen wird der Test auf korrektes Verhalten zunehmend komplexer, zeitaufwendiger und teurer, gleichzeitig bekommt er aber auch eine immer wichtiger werdende wirtschaftliche Bedeutung für das schaltungsherstellende Unternehmen. Diese Erfindung schlägt einen methodischen Vorverarbeitungschritt vor der eigentlichen Verifikationsaufgabe vor. Ziel ist es, durch eine semantikerhaltende Vorverarbeitung ein größenreduziertes Modell der Schaltung zu erhalten, auf dem dann die eigentliche Verifikation erfolgt. Hierbei können bestehende Verifikationsverfahren angewendet werden, die dann i.a. deutlich schneller laufen als auf dem ursprünglichen Design. Die Abstraktion auf das reduzierte Modell erfolgt dabei in solch einer Weise, daß die Verifikationsergebnisse auf dem reduzierten Modell übertragbar auf das Original sind.

## 1 Einleitung

Register-Transfer-Level (RTL) Beschreibungen einer digitalen Schaltung enthalten ein größeres Maß an expliziter Information als Bit-Level Beschreibungen. Auf Bit-Level (wie z.B. in Gatterlisten) sind alle Signale genau ein Bit breit, als Operatoren stehen nur Boolesche Gatter zur Verfügung. Im Gegensatz dazu sind auf RT-Level Strukturen wie Bitvektoren und Busse ebenso sichtbar wie High-Level Operatoren (z.B. Addierer, Shifter, Multiplexer). Beispiele für RT-Level Beschreibungen sind u.a. Hardwarebeschreibungssprachen wie VHDL und Verilog. Ziel des hier beschriebenen Ansatzes ist es, dieses Maß an RT-Information einer Schaltungsbeschreibung auszunutzen, um die Verifikationsaufgabe für Hardware-Verifikations-Tools zu vereinfachen und zu beschleunigen. Die Erfindung betrifft eine neue Abstraktionstechnik, die als Präprozeß im High-Level Property Checking für digitale Schaltungen vorgeschlagen wird.

Sind eine RTL Spezifikation einer Schaltung und eine Eigenschaftbeschreibung gegeben, so berechnet das Verfahren ein reduziertes Modell der Schaltung, auf dem dann die eigentliche Verifikation durchgeführt wird. In dem reduzierten Modell ist die Breite eines jeden Signals auf die jeweils minimale Anzahl an Bits verringert, die nötig ist, um noch in der Lage zu sein, die Gültigkeit der Eigenschaft für die Schaltung zu beweisen oder zu widerlegen. Die generelle Datenfluß- und Kontrollfluß-Struktur der Schaltung bleibt dabei erhalten. Weiteres wesentliches Charakteristikum der Modellreduktion ist, daß die Abstraktion in einer Weise erfolgt, so daß die zu untersuchende Eigenschaft genau dann für die Schaltung gilt, wenn sie für das reduzierte Modell gilt. Die Verifikation kann also vollständig auf dem verkleinerten Modell vorgenommen werden, ein False-Negative (in dem Sinne, daß die Eigenschaft in einem der Modelle gilt und in dem anderen nicht) kann nicht auftreten und muß somit nicht überprüft werden. Weiterhin stellt das Verfahren eine Methode zur Verfügung, die im Fall, daß die Eigenschaft ungültig ist, aus einem Gegenbeispiel, das auf dem reduzierten Modell gefunden wird, ein Gegenbeispiel für die ursprüngliche Schaltung berechnet. Somit erlaubt es das Verfahren, den gesamten Abstraktionsprozeß für einen Benutzer unsichtbar zu machen.

Im allgemeinen laufen Verifikationsprozeduren auf kleineren Modellen schneller als auf großen. Das Verfahren eröffnet die Möglichkeit, bestehende Schaltunggrößen schneller zu verifizieren bzw. größere Schaltungen als vorher mit bestehenden Techniken möglich mit eben denselben Techniken zu bearbeiten oder komplexere und kompliziertere Eigenschaften zu überprüfen. Das reduzierte Modell ist dabei selber wieder eine RTL-Spezifikation einer skalierten Version der Ausgangsschaltung. Vorverarbeitung und eigentliche Verifikation erfolgen voneinander unabhängig und getrennt, was eine Kombination des hier vorgestellten Verfahrens mit einer Bandbreite an bestehenden Verifikationsverfahren zuläßt.

## 2 Das technische Problem, das durch die Erfindung gelöst wird

Diese Erfindung betrifft einen Ansatz und eine Methode, um durch eine semantikerhaltende Vorverarbeitung ein größenreduziertes Modell einer Schaltung zu erhalten, auf dem dann bestehende Verifikationsverfahren angewendet werden. Der Anspruch der Erfindung ist es, ein neues Verfahren für eine solche Modellreduzierung geschaffen zu haben, das folgendes leistet:

- Die zu verifizierende Eigenschaft gilt genau dann für das Ausgangsmodell, wenn sie für das verkleinerte Modell gilt.
- Gilt eine Eigenschaft nicht, und ein Hardware-Verifikations-Tool liefert ein Gegenbeispiel in Form von Eingabebelegungen für das reduzierte Modell, so läßt sich hieraus unmittelbar ein Gegenbeispiel in Form von Eingabebelegungen für die ursprüngliche Schaltung generieren.
- Ursprüngliche Schaltung und reduziertes Modell unterscheiden sich voneinander nur in den (Bit-) Breiten der Signale und Busse.
- Die Größe des reduzierten Modells ist in Bezug auf Schaltung, Eigenschaft und das hier vorgestellte Verfahren minimal.

Das Anwendungsszenario des Verfahrens ist dabei wie folgt charakterisiert:

- Verifikation wird im Sinne von Bounded Model Checking verstanden.
- Die Verifikationsaufgabe wird als Überprüfung einer Eigenschaft für eine gegebene Schaltung formuliert.
- Die Schaltung liegt in einer High-Level Beschreibung vor (z.B. Hardwarebeschreibungsprache wie VHDL oder Verilog, RTL-Netzliste, etc.).
- Die Eigenschaft beschreibt das zu überprüfende Verhalten der Schaltung innerhalb eines endlichen Zeitintervalles.

Folgende Vorteile ergeben sich bei einem solchen Vorgehen:

- Auf dem reduzierten Modell können herkömmliche Verifikationsverfahren, die auf der ursprünglichen Schaltung angewendet wurden, weiter angewendet werden, es ist keine spezielle Abstimmung dieser Verfahren auf das reduzierte Modell nötig.
- Bei der Verifikation auf dem reduzierten Modell kann es in den Ergebnissen nicht zu False-Negatives kommen wie es bei bestimmten anderen Abstraktionstechniken möglich ist. Gegenbeispiele müssen nicht weiter auf Echtheit überprüft werden und können direkt in Form von Gegenbeispielen für die Ausgangsschaltung dargestellt werden.
- Das Verfahren analysiert die Skalierbarkeit der Schaltung und berechnet eine maximale Skalierung, d.h. ein minimales Modell.
- Da die Modellreduzierung lediglich bzgl. der Skalierung von Signalbreiten erfolgt, bleibt die generelle Schaltungsstruktur komplett erhalten.
- Da das Laufzeitverhalten vieler HW-Verifikationsverfahren i.a. exponentiell ist, birgt eine Größenreduzierung durch Skalierung ein enormes Potential zur Laufzeitbeschleunigung.
- Skalierung als Vorverarbeitung birgt weiterhin das Potential, Schaltungen zu bearbeiten, deren Größen bislang die Grenzen und Kapazitäten bestehender Tools überschreiten.
- Ist für eine Schaltung bedingt durch ihre Charakteristika keine Reduzierung möglich, so ist das Ergebnis des Verfahrens (das reduziertes Modell) gleich der Ausgangsschaltung. Es tritt in diesem Sinne niemals eine Verschlechterung der Verifikationsaufgabe durch Benutzung des Verfahrens als Präprozeß ein.
- Untersuchungen haben gezeigt, daß die Laufzeiten für dieses Verfahren und die Vorverarbeitung gegenüber der Laufzeit der anschließenden eigentlichen Verifikation komplett zu vernachlässigen sind.
- Insbesondere läßt sich beschriebenes Verfahren und beschriebene Methodologie auch beim High-Level Equivalence Checking anwenden, was als ein Spezialfall des oben beschriebenen Anwendungsszenarios angesehen werden kann. Hierbei kann z.B. die Schaltungsbeschreibung zwei verschiedene High-Level Implementierungen desselben Designs beinhalten und die Eigenschaftsbeschreibung die Äquivalenz beider Implementierungen fordern. Oder aber die Eigenschaftsbeschreibung kann selber eine funktionale Spezifikation der Schaltung sein, z.B. auch in einer Hardwarebeschreibungsprache.

## 3 Das Problem-Umfeld

Abbildung 1 zeigt, wie sich das Verfahren ins klassische Property Checking Framework einfügt. Gegeben sei die Beschreibung eines digitalen Designs in einer Hardwarebeschreibungssprache, wie z.B. Verilog oder VHDL. Zunächst erfolgt eine Übersetzung in eine RTL-Netzliste aus High-Level Operatoren, wie z.B. Addierern, Multiplexern, Booleschen vektorwertigen Gattern und Vergleichern. Alle Signale haben eine fest vorgegebene Bitbreite und

nehmen Vektoren aus Bits als Werte an. Signale auf dem Kontroll-Pfad haben typischerweise die Breite 1, und Signale auf dem Datenpfad haben z.B. je nach Anwendungsgebiet der Schaltung Breiten von 4 bis zu 64 Bits. Property-Checker nehmen solch eine RTL Netzliste und eine Eigenschaftsspezifikation (in der Regel in einer formalen Eigenschaftssprache) als Eingabe und beweisen die Gültigkeit dieser Eigenschaft, bzw. geben andernfalls ein Gegenbeispiel zurück.

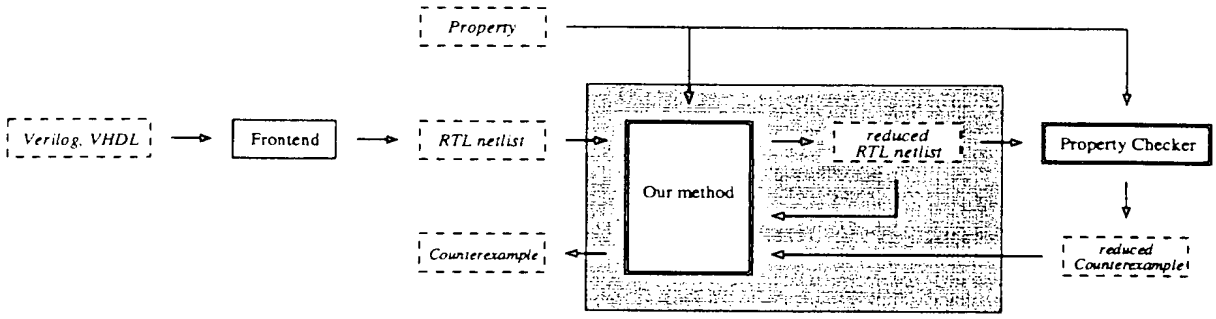



**Abbildung1.** Property Checking Flow

Das Verfahren nimmt eine RTL Netzliste und eine Eigenschaftsbeschreibung und berechnet ein verkleinertes Modell der Schaltung, in dem alle Signalbreiten auf ein notwendiges Minimum herunterskaliert sind. Dieses abstrahierte Modell wird an Stelle des Originals als Eingabe an die Beweiser weitergereicht. Die Abstraktions liefert eine eigenschaftsspezifisches 1-zu-1 Modell der RTL Netzliste. Dies bedeutet, daß die Eigenschaft genau dann für das Ausgangsdesign gilt, wenn sie auch für das reduzierte Modell gilt. Somit können keine falschen Gegenbeispiele auftreten. Gilt die gegebene Eigenschaft für das Design nicht, gibt der Property Checker ein Gegenbeispiel zurück, das Wertebelegungen für alle Eingangssignale des reduzierten Modells angibt. Das Verfahren nimmt solch ein Gegenbeispiel und berechnet daraus ein Gegenbeispiel für die Originalschaltung. Die 1-zu-1 Abstraktion des Verfahrens garantiert, daß es nur genau dann ein Gegenbeispiel, das die Eigenschaft verletzt, für das eine Modell gibt, wenn es auch ein Gegenbeispiel für das andere Modell gibt.

Die Verfahren zur Generierung des abstrahierten Modells und für die evtl. Rückrechnung von Gegenbeispielen sind strikt getrennt vom eigentlichen Model Checking. Das Verfahren ist unabhängig von der konkreten Realisierung des Property Checkers und läßt sich so mit einer Vielzahl verschiedener Beweisverfahren kombinieren, die RTL Beschreibungen als Eingabe nehmen, unabhängig davon ob der zugrundeliegende Beweiser auf Bit-Ebene arbeitet (z.B. SAT-Verfahren oder BDD basierte Methoden) oder High-Level Techniken benutzt (ILP-Verfahren, arithmetische Constraint-Solver).

## 4 Lösungsansatz dieser Erfindung

Das vorgestellte Verfahren benutzt eine Kombination aus Techniken zur Analyse von strukturellen Datenpfad-Abhängigkeiten und interpretierten und uninterpretierten Abstraktionen, um die Signalbreiten in RTL Design Modellen herunter zu skalieren. Die grundlegende Idee

ist es, die Bitbreite eines jeden im Design vorkommenden Signals auf die minimale Anzahl von Bits zu reduzieren, die ausreichend ist, um die Eigenschaft überprüfen zu können.

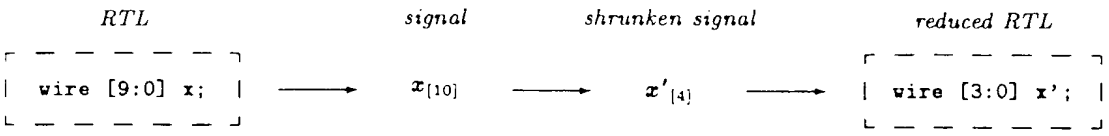



**Abbildung2.** Signalbreiten-Skalierung/Reduzierung

Als erster Schritt wird die RTL Netzliste in ein äquivalentes System aus Gleichungen in einer formalen Bitvektor-Theorie übersetzt. Dieses Gleichungssystem ist genau dann lösbar (d.h. erfüllbar), wenn die gegebene Eigenschaft **nicht** für das Design gilt. Eine Lösung des Gleichungssystems, so sie existiert, liefert ein Gegenbeispiel, das für jedes Signal des Designs eine Wertebelegung angibt, so daß die Eigenschaft unter der Gesamtheit dieser Wertebelegungen verletzt ist.

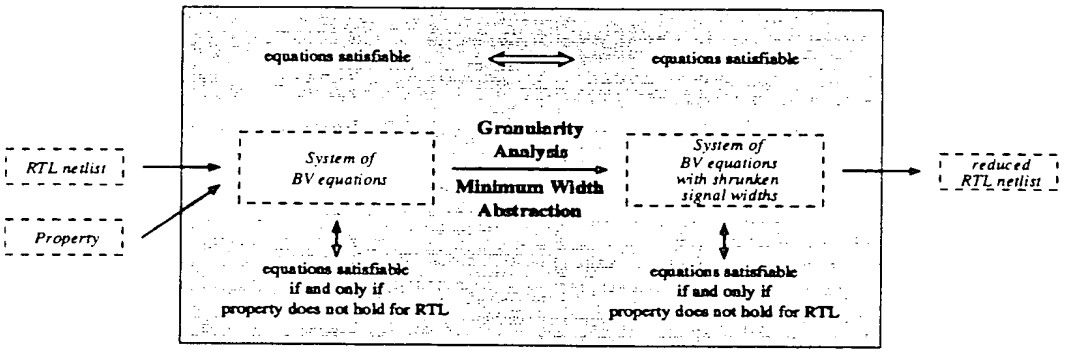



**Abbildung3.** Generierung des reduzierten Modells

Die Leistung des hier vorgestellten Verfahren ist die Konstruktion eines abstrakten Modells dieses Gleichungssystems. Die strukturellen Datenabhängigkeiten innerhalb der Bitvektor-Gleichungen werden analysiert und ein zweites Gleichungssystem wird berechnet, in dem die Breite einer jeden Bitvektor-Variablen auf die geringstmögliche Anzahl Bits reduziert ist, so daß das erhaltene Gleichungssystem genau dann erfüllbar ist, wenn auch das Ausgangsgleichungssystem erfüllbar ist.

- Die Datenabhängigkeiten der einzelnen Signale der Schaltung untereinander und die Abhängigkeiten von zusammenhängenden Bereichen innerhalb der Signale werden analysiert.
- Für jedes Signal wird eine Unterteilung in zusammenhängende Signalbereiche (Granularität) berechnet, in der jeweils die Bits eines solchen Bereiches uniform behandelt werden können.

- Basierend auf den Ergebnissen der beiden vorhergehenden Schritte wird für jeden Bereich der Granularität eines Signals die minimale Anzahl an Bits berechnet, auf die die Breite dieses Bereiches reduziert werden kann.
- Zur Generierung des reduzierten Modells wird in der Ausgangsschaltung jedes Signal auf die hierfür vorhergehend berechnete minimale Breite reduziert (genauer: auf die Summe der minimalen Breiten der einzelnen Bereiche).
- Das Verfahren garantiert die Äquivalenz von Gültigkeit der Eigenschaft auf der ursprünglichen Schaltung und Gültigkeit auf dem reduzierten Modell.
- Das Verfahren berechnet die maximal mögliche Skalierung, d.h. die minimal möglichen Signalbreiten, die ausreichend für die Eigenschaftsüberprüfung auf Schaltung und reduziertem Modell ist.

Die Modell-Generierung unterteilt sich in zwei aufeinanderfolgende Phasen. Zunächst wird für jede Bitvektor-Variable die gröbstmögliche Granularität berechnet, wie sie von den strukturellen Datenabhängigkeiten innerhalb der Bitvektor-Gleichungen vorgegeben wird. Unter einer **Granularität** versteht man dabei eine Aufteilung eines Bitvektors in verschiedene zusammenhängende Stücke (*Chunks*), die die jeweils gröbstmögliche Zusammenfassung einzelner Bits innerhalb des Vektors darstellen. Diese wird in Abschnitt 4.2 veranschaulicht.

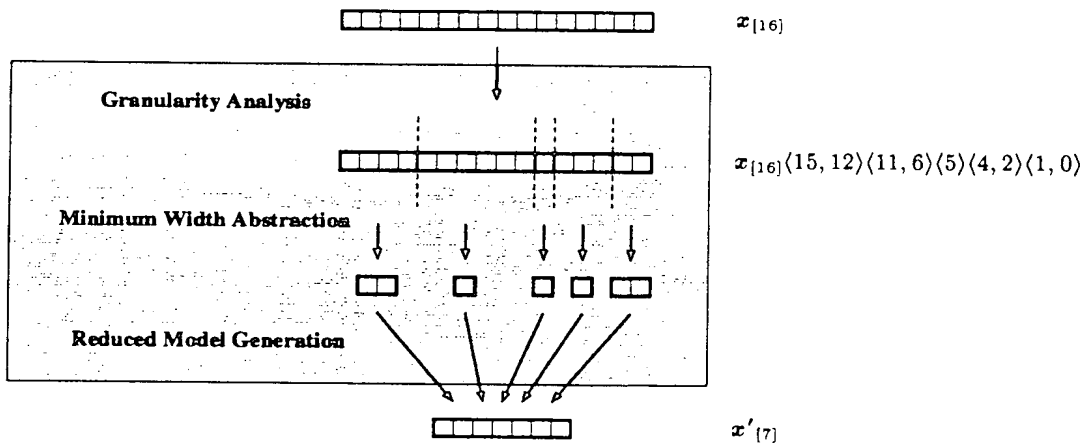



**Abbildung4.** Abstraktionstechnik

Anschließend wird für jeden Chunk die notwendige minimale Breite berechnet, siehe auch Abbildung 4.3. Gemäß dieser berechneten minimalen Breiten wird ein reduziertes Modell jeder Bitvektorvariablen zusammengesetzt. Das reduzierte System von Bitvektorgleichungen unterscheidet sich von dem Original lediglich in den Breiten der Variablen, wobei die Breite einer jeden Variablen im reduzierten System kleiner oder gleich der Breite der entsprechenden Variable im Ausgangsgleichungssystem ist. Ein weiterer Beitrag des Verfahrens ist eine Methode zur Berechnung von Lösungen für das Originalgleichungssystem aus Lösungen des reduzierten Systems. Diese Berechnung basiert auf den Ergebnissen der vorherigen Granularitätsanalyse und Minimale-Breiten-Berechnung.

### 4.1 Eine formale Theorie von Bitvektoren fester Breite

Wir definieren eine gleichungsbasierte formale Theorie von Bitvektoren fester Breite, die eine Erweiterung der Kerntheorie von Bitvektoren, die in [Moe98] angegeben wird, darstellt. $\mathbb{B} := \{0, 1\}$ bezeichne die Menge von *Bit-Werten* 0 und 1. Ein *Bitvektor* der Breite $n \in \mathbb{N}_+$ ist ein Vektorelement von $\mathbb{B}^n$, bestehend aus $n$ einzelnen Bits, die von rechts nach links beginnend mit 0 durchindiziert werden. $\mathbb{B}^n$ sei durch $\mathbb{B}_{[n]}$ bezeichnet.

**Definition 1 (Bitvektorvariablen).** *Sei $n \in \mathbb{N}_+$. Eine* Bitvektorvariable $x_{[n]}$ *der Breite $n$ ist eine getypte Variable, die einen Bitvektor $v \in \mathbb{B}_{[n]}$ der festen Breite $n$ repräsentiert.* ⊠

Für jede Bitvektorvariable ist ihr Typ, d.h. ihre Breite $n$, eine beliebige aber fixe wohlbekannte positive natürliche Zahl. Desweiteren beinhaltet unsere Theorie Bitvektorkonstanten $c_{[m]}$, $m \in \mathbb{N}_+$ und $c \in \mathbb{N}$, die als binäre Bitstrings notiert werden. Im Unterschied zur Kerntheorie von [Moe98], die lediglich Extraktion und Konkatenation als Operatoren kennt, ist unsere Theorie um einen kompletten Satz von High-Level Operatoren erweitert, deren Zusammenfassung in Abbildung 5 zu sehen ist. Neben bitweisen Booleschen Operatoren sind z.B. Arithmetik und Speicherzugriffe modelliert, und insbesondere ein if-then-else Operator,[1] der die Beschreibung von dynamischen Datenabhängigkeiten gestattet.

| Bitvector Operator | Syntax | Example |
|---|---|---|
| bitvector variables<br>bitvector constants | $x_{[n]}$<br>$c_{[m]}$ | $x_{[8]}, y_{[1]}, z_{[4]}, \ldots$<br>$10011_{[5]}, 00111111_{[8]}, 0_{[1]}, 1_{[1]}, \ldots$ |
| concatenation | $\otimes$ | $x_{[16]} \otimes y_{[4]}$ |
| extraction | $[j, i]$ | $x_{[8]}[5, 2]$ |
| negation | neg | $\mathsf{neg}(x_{[8]})$ |
| bitwise Boolean operations | and, or, nor<br>nand, nor, xnor | $x_{[12]}$ and $y_{[12]}$, $x_{[12]}$ or $y_{[12]}$, $x_{[12]}$ xor $y_{[12]}$<br>$x_{[12]}$ nand $y_{[12]}$, $x_{[12]}$ nor $y_{[12]}$, $x_{[12]}$ xnor $y_{[12]}$ |
| Boolean reductions | redAnd, redOr, redXor | $\mathsf{redAnd}(x_{[8]})$, $\mathsf{redOr}(x_{[8]})$, $\mathsf{redXor}(x_{[8]})$ |
| if-then-else | ite | $\mathsf{ite}(a_{[4]} = b_{[4]}, x_{[8]}, y_{[8]})$<br>$\mathsf{ite}(a_{[4]} < b_{[4]}, x_{[8]}, y_{[8]})$ |
| arithmetics | $+$<br>$*$ | $x_{[32]} + y_{[32]}$<br>$x_{[16]} * y_{[16]}$ |
| memory read<br>memory write | $mem_{[m \cdot n]}[i_{[l]}]$ | $x_{[10]} := mem_{[128 \cdot 10]}[i_{[7]}]$<br>$mem_{[128 \cdot 10]}[i_{[7]}] := x_{[10]}$ |

**Abbildung5.** Eine formale Theorie von Bitvektoren fester Breite

Die Menge wohldefinierter Bitvektor-Therme ist in der üblichen Art definiert, Wohldefiniertheit impliziert, daß Variablenbreiten mit den entsprechenden Operatoren verträglich sind und daß Extraktionen nicht die Breiten der zugehörigen Therme über- oder unterschreiten. Der obige Satz an Operatoren ist ausreichend um z.B. komplette VHDL oder Verilog Designs

[1] Die Booleschen Prädikate = und < sind auf Bitvektorargumenten gleicher Breite definiert. Gleichheit ist bitweise definiert, < gemäß der lexikographischen Ordnung auf Bitstrings.

innerhalb dieser Theorie zu beschreiben. Zusätzliche High-Level Operatoren, wie z.B. Shifts, Rotieren oder weitere Boolesche Prädikate, lassen sich mit Hilfe der vorhandenen Operatoren ausdrücken, ohne daß sich dadurch Änderungen bzgl. der Skalierbarkeit ergeben. Variablenbelegungen, Gleichungen von Bitvektor-Thermen und der Erfüllbarkeitsbegriff werden in der offensichtlichen Weise eingeführt.

**Beispiel 1 (Bitvektorgleichungen)** *Seien* $\boldsymbol{x}_{[16]}, \boldsymbol{y}_{[4]}$ *und* $\boldsymbol{z}_{[4]}$ *Bitvektorvariablen.*

$$\begin{array}{lrcl} 1. & \boldsymbol{x}_{[16]}[15,8] \otimes \boldsymbol{x}_{[16]}[7,0] & = & \boldsymbol{x}_{[16]} \\ 2. & \boldsymbol{x}_{[16]} & = & \mathsf{neg}(\boldsymbol{x}_{[16]}) \\ 3. & \boldsymbol{y}_{[4]} \text{ and } 1100_{[4]} & = & \boldsymbol{z}_{[4]} \end{array}$$

*Gleichung 1 ist allgemeingültig und Gleichung 2 unerfüllbar. Die dritte Gleichung ist erfüllbar, z.B. durch* $\boldsymbol{y}_{[4]} := 0111_{[4]}$ *und* $\boldsymbol{z}_{[4]} := 0100_{[4]}$*, sie ist aber nicht allgemeingültig.* □

**Beispiel 2 (Systeme von Bitvektorgleichungen)** *Sei folgendes System von Bitvektorgleichungen mit Bitvektorvariablen* $\boldsymbol{x}_{[8]}$ *und* $\boldsymbol{y}_{[4]}$ *gegeben.*

$$\begin{array}{lrcl} & ( \quad \boldsymbol{x}_{[8]} & = & \boldsymbol{y}_{[4]} \otimes \boldsymbol{y}_{[4]} \quad ) \\ \wedge & ( \boldsymbol{x}_{[8]}[4,4] & = & \mathsf{neg}(\boldsymbol{x}_{[8]}[0,0]) \; ) \end{array}$$

*Einzelnd genommen ist sowohl die erste Gleichung als auch die zweite erfüllbar, das Gleichungssystem in seiner Gesamtheit jedoch ist unerfüllbar.* □

### 4.2 Granularitätsanalyse

Gegeben sei ein System von Bitvektorgleichungen. Das hier vorgestellte Verfahren analysiert in einem ersten Schritt die strukturellen Datenabhängigkeiten aller Bitvektorvariablen untereinander, wie sie von den in den Gleichungen vorkommenden Operatoren bestimmt werden. Gemäß dieser Analyse werden die einzelnen Bitvektorvariablen in *Chunks* zerlegt, und auf diese Weise wird für jede Variable eine *Granularität* berechnet.

**Definition 2 (Chunk).** *Ein* Chunk $\boldsymbol{x}_{[n]}\langle j,i\rangle$, $n > j \geq i \geq 0$, *einer Bitvektorvariablen* $\boldsymbol{x}_{[n]}$ *ist ein zusammenhängender Teilbereich von* $\boldsymbol{x}_{[n]}$*. Chunks sind genau die Resultate von Extraktionsoperationen auf Bitvektorvariablen, d.h.* $\boldsymbol{x}_{[n]}\langle j,i\rangle := \boldsymbol{x}_{[n]}[j,i]$. ■

**Definition 3 (Granularität).** *Eine* Granularität *einer Bitvektorvariablen* $\boldsymbol{x}_{[n]}$ *ist eine vollständige Partitionierung von* $\boldsymbol{x}_{[n]}$ *in verschiedene einzelne Chunks von* $\boldsymbol{x}_{[n]}$*. Chunks* $\boldsymbol{c}_{[m_1]}$, $\boldsymbol{c}_{[m_2]}$, …, $\boldsymbol{c}_{[m_k]}$ *einer Bitvektorvariablen* $\boldsymbol{x}_{[n]}$ *bilden genau dann eine Granularität von* $\boldsymbol{x}_{[n]}$*, wenn gilt* $\boldsymbol{x}_{[n]} = \boldsymbol{c}_{[m_1]} \otimes \boldsymbol{c}_{[m_2]} \otimes \ldots \otimes \boldsymbol{c}_{[m_k]}$. ■

**Beispiel 3** $x_{[16]}\langle 15,\ 8\rangle, x_{[16]}\langle 7,\ 5\rangle, x_{[16]}\langle 4,\ 4\rangle, x_{[16]}\langle 3,\ 0\rangle$ *bilden eine Granularität von* $x_{[16]}$. *Als Notation kürzen wir eine solche Granularität wie folgt ab:* $x_{[16]}\langle 15,8\rangle\langle 7,5\rangle\langle 4,4\rangle\langle 3,0\rangle$. *Ein weiteres Beispiel hierzu findet sich in Abbildung 4.* □

Während der Granularitätsanalyse werden Variablen (oder bereits berechnete Chunks) dann und nur dann in (weitere) Chunks zerlegt, wenn es durch die Operatoren in den Gleichungen erzwungen wird, d.h. wenn es die strukturellen Abhängigkeiten vorschreiben. Strukturelle Datenabhängigkeiten sind z.B. durch Extraktionen, Konkatenationen, Reduktionen, Arithmetik und Speicherzugriffe bedingt. Im allgemeinen treten sie immer dann auf, wenn eine Bitvektorvariable oder ein Term nicht als Einheit behandelt wird, sondern verschiedene Operationen auf unterschiedlichen Bereichen ausgeführt werden. Diese strukturellen Abhängigkeiten können dynamisch bedingt sein, z.B. wenn im then- und else-Fall einer Fallunterscheidung unterschiedliche Abhängigkeiten auftreten.

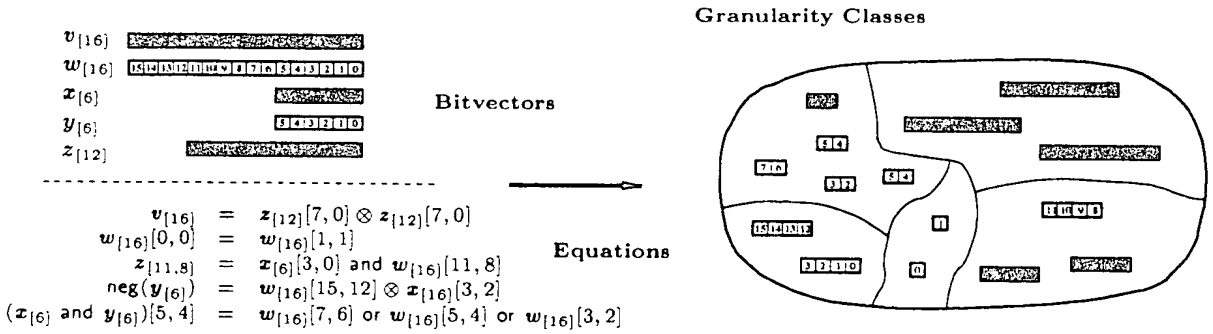



**Abbildung6.** Granularitätsanalyse

Das Verfahren berechnet für jede Bitvektorvariable die gröbstmögliche Granularität, die sowohl die statischen als auch dynamischen Datenabhängigkeiten berücksichtigt. Die Berechnung erfolgt unter Verwendung einer speziellen dynamischen Äquivalenzklassen-Struktur, wobei Bitvektorvariablen bzw. Chunks immer genau dann weiter zerlegt werden, wenn während der Analyse auf strukturelle Extraktionen getroffen wird. Zwei Chunks zweier (nicht notwendigerweise verschiedener) Bitvektorvariablen landen genau dann in derselben Äquivalenzklasse, wenn ihre einzelnen Bitpositionen durch die Opertoren einer Gleichung bitweise zueinander in Beziehung gesetzt werden. Auf die Weise werden wechselseitig strukturell abhängige Chunks in Äquivalenzklassen gruppiert, veranschaulicht in Abbildung 6.

**Beispiel 4** *Man betrachte folgende Bitvektorgleichung:* $z_{[8]} = x_{[4]} \otimes y_{[4]}$. *Initial liegen alle Bitvektorvariablen in verschiedenen Äquivalenzklassen, nämlich :* $\{x_{[4]}\}$, $\{y_{[4]}\}$, $\{z_{[8]}\}$. *Der Konkatenationsoperator in der rechten Seite der Gleichung erfordert eine Zerlegung von* $z_{[8]}$ *zwischen den Bitpositionen 3 und 4, und die Gleichung selbst erzwingt, daß der obere Bereich von* $z_{[8]}$ *strukturell äquivalent zu* $x_{[4]}$ *und der untere Bereich zu* $y_{[4]}$ *ist. Somit ist der Folgezustand der Äquivalenzklassen* $\{z_{[8]}\langle 7,4\rangle, x_{[4]}\}$, $\{z_{[8]}\langle 3,0\rangle, y_{[4]}\}$. □

Die Äquivalenzklassen-Berechnung erfolgt inkrementell. Jede Gleichung wird genau einmal von dem Verfahren analysiert und der dadurch bedingte Folgezustand der Äquivalenzklassen berechnet. Klassen werden, falls nötig, vereinigt oder zerlegt, was in dieser speziellen Struktur bedeutet, daß alle Chunks innerhalb einer Klasse an derselben Bitposition getrennt werden und die entsprechenden Teile in zwei neuen Klassen gruppiert werden. Sobald alle Gleichungen des Systems auf diese Weise analysiert worden sind, ist für jede Bitvektorvariable die gröbstmögliche Granularität durch den Zustand der Äquivalenzklassen gegeben, d.h. durch die Breiten der zugehörigen Chunks in den Klassen.

### 4.3 Minimale-Breiten-Abstraktion

Im Anschluß an die Granularitätsanalyse wird für jede Äquivalenzklasse $C$ getrennt die minimale Chunk-Breite $\varphi(C)$ berechnet, für die gilt, daß das Ausgangssytem von Bitvektorgleichungen genau dann erfüllbar ist, wenn das reduzierte System erfüllbar ist, in dem die Breite aller Chunks aus $C$ auf $\varphi(C)$ reduziert ist.

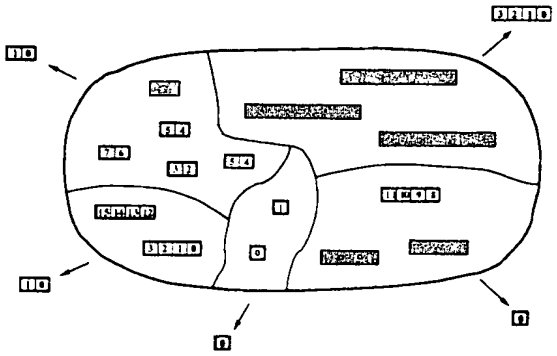


**Abbildung7.** Minimale-Breiten Berechnung

$\varphi(C)$ ist eine Funktion, deren Ergebnis sowohl von der Anzahl Chunks in $C$ abhängt als auch von Anzahl und Art der verschiedenen High-Level Operatoren, die in den Bitvektorgleichungen auf den Chunks aus $C$ operieren. Details über die genaue Berechnung von $\varphi(C)$ würden den Rahmen dieses Berichtes sprengen, stattdessen werden einige prinzipielle Beispiele gegeben.

**Beispiel 5** *Für Gleichungen, die lediglich Konkatenation, Extraktion und Negation als Operatoren enthalten, kann die Breite aller Chunks in einer Klasse auf ein Bit reduziert werden.*

$$\boldsymbol{x}_{[16]} = \boldsymbol{y}_{[8]} \otimes \boldsymbol{z}_{[8]} \quad \overset{erfüllbar}{\Longleftrightarrow} \quad \boldsymbol{x'}_{[2]} = \boldsymbol{y'}_{[1]} \otimes \boldsymbol{z'}_{[1]}$$

*Aus einer Lösung für die reduzierte rechte Gleichung, läßt sich eine Lösung für die linke Ausgangsgleichung durch Kopieren des Wertes von $\boldsymbol{y'}_{[1]}$ in alle Bits von $\boldsymbol{y}_{[8]}$ und alle von $\boldsymbol{x}_{[16]}[15,8]$ und durch Kopieren von $\boldsymbol{z'}_{[1]}$ in alle Bits von $\boldsymbol{z}_{[8]}$ und $\boldsymbol{x}_{[16]}[7,0]$ erhalten.* □

**Beispiel 6** *Für Gleichungen, die lediglich bitweise Boolesche Operationen enthalten, kann die Breite aller Chunks in einer Klasse auf ein Bit reduziert werden.*

$$z_{[8]} = x_{[8]} \text{ and } y_{[8]} \quad \overset{erfüllbar}{\Longleftrightarrow} \quad z'_{[1]} = x'_{[1]} \text{ and } y'_{[1]}$$

*Wiederum läßt sich eine Lösung für die linke Gleichung durch Kopieren der Bitwerte einer Lösung der rechten Gleichung in alle Bits der entsprechenden Variablen der linken Gleichung berechnen.* □

**Beispiel 7** *Gleichungen, die dynamische Datenabhängigkeiten beinhalten, z.B. if-then-else Konstrukte, erfordern eine Analyse aller möglichen Ungleichungen zwischen Variablen, resultierend aus den Konditionalteilen.*

$$\begin{array}{rcl} a_{[1]} & = & \mathsf{ite}(x_{[8]} = y_{[8]},\ 0_{[1]},\ 1_{[1]}) \\ b_{[1]} & = & \mathsf{ite}(y_{[8]} = z_{[8]},\ 0_{[1]},\ 1_{[1]}) \\ c_{[1]} & = & \mathsf{ite}(z_{[8]} = x_{[8]},\ 0_{[1]},\ 1_{[1]}) \\ 1_{[1]} & = & a_{[1]} \text{ and } b_{[1]} \text{ and } c_{[1]} \end{array} \quad \overset{erfüllbar}{\Longleftrightarrow} \quad x_{[8]} \neq y_{[8]} \ \wedge\ y_{[8]} \neq z_{[8]} \ \wedge\ z_{[8]} \neq x_{[8]}$$

*Es gilt:*

$$\begin{array}{rcl} x_{[8]} \neq y_{[8]} & & x'_{[1]} \neq y'_{[1]} \\ \wedge\ y_{[8]} \neq z_{[8]} & \overset{erfüllbar}{\Leftrightarrow} & \wedge\ y'_{[1]} \neq z'_{[1]} \\ \wedge\ z_{[8]} \neq x_{[8]} & & \wedge\ z'_{[1]} \neq x'_{[1]} \end{array}$$

*Die Minimale-Breiten Berechnung des hier vorgestellten Verfahrens liefert*

$$\begin{array}{rcl} x_{[8]} \neq y_{[8]} & & x'_{[2]} \neq y'_{[2]} \\ \wedge\ y_{[8]} \neq z_{[8]} & \overset{erfüllbar}{\Longleftrightarrow} & \wedge\ y'_{[2]} \neq z'_{[2]} \\ \wedge\ z_{[8]} \neq x_{[8]} & & \wedge\ z'_{[2]} \neq x'_{[2]} \end{array}$$

*und somit folgende Reduzierung des Gleichungssystems:*

$$\begin{array}{rcl} a_{[1]} & = & \mathsf{ite}(x_{[8]} = y_{[8]},\ 0_{[1]},\ 1_{[1]}) \\ b_{[1]} & = & \mathsf{ite}(y_{[8]} = z_{[8]},\ 0_{[1]},\ 1_{[1]}) \\ c_{[1]} & = & \mathsf{ite}(z_{[8]} = x_{[8]},\ 0_{[1]},\ 1_{[1]}) \\ 1_{[1]} & = & a_{[1]} \text{ and } b_{[1]} \text{ and } c_{[1]} \end{array} \quad \overset{erfüllbar}{\Longleftrightarrow} \quad \begin{array}{rcl} a'_{[1]} & = & \mathsf{ite}(x'_{[2]} = y'_{[2]},\ 0_{[1]},\ 1_{[1]}) \\ b'_{[1]} & = & \mathsf{ite}(y'_{[2]} = z'_{[2]},\ 0_{[1]},\ 1_{[1]}) \\ c'_{[1]} & = & \mathsf{ite}(z'_{[2]} = x'_{[2]},\ 0_{[1]},\ 1_{[1]}) \\ 1_{[1]} & = & a'_{[1]} \text{ and } b'_{[1]} \text{ and } c'_{[1]} \end{array}$$

□

Das in dieser Erfindungsmeldung vorgestellte Verfahren ist in der Lage, die Minimale-Breiten Berechnung für Gleichungssysteme durchzuführen, die den kompletten Operatorsatz der in Abschnitt 4.1 vorgestellten Bitvektor-Theorie benutzen. Dabei können die Operatoren in beliebiger Klammerung oder Verschränkung in den Thermen der Gleichungen benutzt werden. Sind die minimalen Breiten für alle Äquivalenzklassen berechnet, so erfolgt die Generierung des reduzierten Gleichungssystems durch Ersetzung der Breiten der Chunks der Variablen im Ausgangsgleichungssystem durch die berechnete minimale Breite der zu einem Chunk gehörigen Äquivalenzklasse.

### 4.4 Zusammenfassung

Abschließend werden die Methoden, die diese Erfindung zur Verfügung stellt, zusammengefaßt:

- Verfahren zur **Granularitäts-Analyse** für RTL Schaltungsbeschreibungen mit komplettem RTL-Operator Satz
- Verfahren für **Minimale-Breiten-Abstraktion** basierend auf RTL Granularitätsanalyse
- Granularitäts-Analyse und Minimale-Breiten-Abstraktion zur Berechnung eines größenreduzierten 1-zu-1 Modells von RTL Schaltungsbeschreibungen
- **Verifikationsmethodologie** zur Anwendung von Granularitätsanalyse und Minimale-Breiten-Abstraktion als **Präprozeß** im (Bounded) Model Checking
- Verfahren zur **Rückrechnung abstrahierter Gegenbeispiele in Gegenbeispiele für das Ausgangsmodell** basierend auf den Ergebnissen der vorherigen Granularitäts-Analyse und Minimale-Breiten-Abstraktion

Besondere Schwierigkeiten, die hierbei überwunden werden mußten bzw. für die eine neue Lösung gefunden wurde, sind:

- Die Granularitätsanalyse kann mit dem **kompletten** Satz an High-Level Operatoren umgehen. Insbesondere berücksichtigt die Analyse nicht nur statische, sondern auch **dynamische Datenabhängigkeiten** (z.B. durch Kontrollstrukturen bedingte Abhängigkeiten wie bei `if-then-else`).
- Die Berechnung der minimalen Signalbreiten im Anschluß an die Granularitätsanalyse stellt eine **neue Abstraktiontechnik** dar.
- Granularitätsanalyse und Minimale-Breiten-Berechnung stellen zu jedem Zeitpunkt sicher, daß die **Eigenschaft genau dann für das reduzierte Modell gilt, wenn sie auch für die ursprüngliche Schaltung gilt.**

## 5 Anwendungsbeispiel

Ein Prototyp des vorgestellten Verfahrens ist implementiert worden und bei ZT-SE-4 von Siemens München und bei Computer Network Peripherals von Infineon in San Jose, Kalifornien, getestet worden. Die Resultate zeigten sich vielversprechend und werden in einer Fallstudie anhand der Adreß-Management-Einheit eines ATM-Switching-Elementes veranschaulicht.

Die Schaltung besteht aus ca. 3000 Zeilen Verilog-Code, die synthetisierte Netzliste umfaßt ungefähr 22.000 Gatter und 35.000 RAM Speicherzellen. Das Design besteht im wesentlichen aus 16 FIFO-Queue Puffern und einer komplexen Kontroll-Logik. Speicheradressen werden über 33 Eingabekanäle an die Management-Einheit gereicht, in den FIFO's gespeichert, und auf Anfrage auf einem von 17 Ausgabekanälen wieder ausgegeben, wobei die Reihenfolge der übergebenen Datenpakete erhalten bleiben soll und keine Adresse aus der Management-Einheit verlorengehen darf. Abbildung 8 zeigt ein Blockdiagramm der Schaltung.

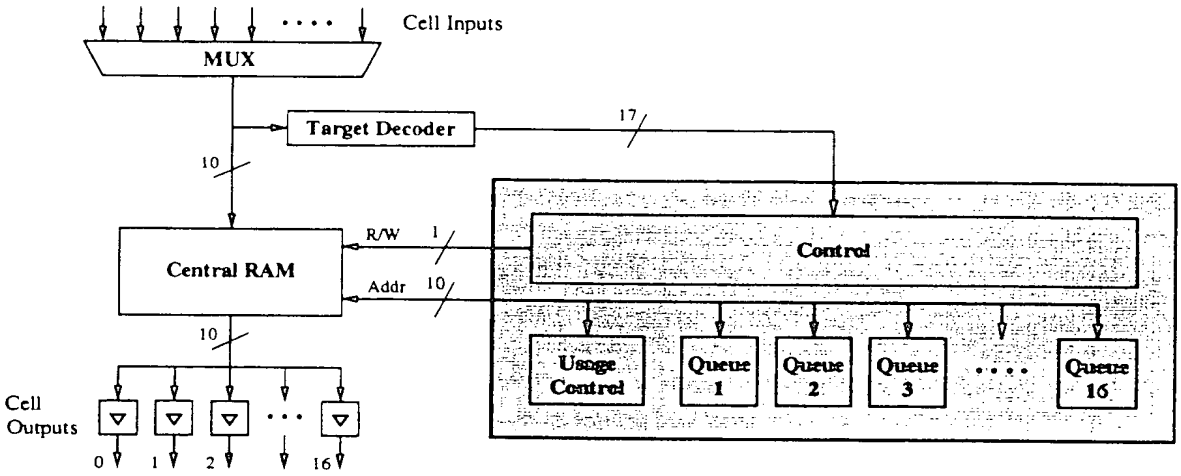



**Abbildung8.** Blockdiagramm des ATM Switching Elements

Der Prototyp wurde als Präprozessor für einige der Siemens Property Checker benutzt. Drei verschiedene Eigenschaften, `nop`, `read` und `write`, sollten verifiziert werden, die jeweils das zu überprüfende Verhalten der Schaltung über 4 (`nop`, `write`) bzw. 6 (`read`) Zeitschritte beschrieben. Das Verfahren analysierte das gegebene Design und die Eigenschaften und generierte reduzierte Modelle der Schaltung, die dann als Eingabe an die Property Checker weitergegeben wurden. Es stelle sich heraus, daß die `write` Eigenschaft nicht erfüllt war aufgrund eines Design-Fehlers im Verilog Code. Die Eigenschaftsverletzung wurde von dem Property Checker erkannt und in Form eines Gegenbeispieles für das reduzierte Modell ausgegeben. Hierdraus wurde ein Gegenbeispiel für die Ausgangsschaltung berechnet und der Fehler im Design behoben, woraufhin die Eigenschaft erneut auf der korrigierten Schaltung und ihrem reduzierten Modell verifiziert wurde (`write_fail`, `write_hold`). Die Laufzeiten des Property Checkers auf den reduzierten Modellen wurden gegen die Laufzeiten auf dem ursprünglichen Design (ohne Präprozess) verglichen und finden sich in Abbildung 5.

Es stellte sich ein erhebliche Reduktion der Modellgrößen und eine erhebliche Beschleunigung der Laufzeiten des Property Checkers heraus. Die Modellgrößen konnten durchschnittlich auch ca. 30% der ursprünglichen Schaltungsgröße (gemessen in der Gesamtzahl an Bits) skaliert werden, die Laufzeiten gingen von zwischen einer halben und einer dreiviertel Stunde auf Minuten bzw. sogar Sekunden zurück. Alle Beispiele wurden auf einem Intel Pentium II PC mit 450 MHz CPU, 128M Hauptspeicher unter Linux gemessen. Insbesondere fällt auf,

daß die Berechnungszeiten des Verfahrens zur Analyse des Designs und zur Generierung des reduzierten Modells von zwischen 3 und 7 Sekunden nahezu vernachlässigbar sind.

| | Eigenschaft | ursprüngliches Design | reduziertes Modell |
|---|---|---|---|
| Laufzeit für Analyse und Generierung des reduzierten Modells | nop | | 2.96 secs |
| | read | | 6.53 secs |
| | write | | 3.24 secs |
| FIFO Größen | nop | 160 * 10 bit | 160 * 2 bit |
| | read | 160 * 10 bit | 160 * 3 bit |
| | write | 160 * 10 bit | 160 * 3 bit |
| Gesamtzahl an Bits | nop | 20925 | 5034 |
| | read | 31452 | 10592 |
| | write | 14622 | 5163 |
| Gesamtzahl an Gattern | nop | 23801 | 5661 |
| | read | 23801 | 7929 |
| | write | 23801 | 7929 |
| Größe der Netzlisten | nop | 150612 | 37001 (24.5 %) |
| | read | 150612 | 51626 (34.3 %) |
| | write | 150612 | 51598 (34.3 %) |
| Property Checker Laufzeiten | nop | 23:33 min | 37.96 secs |
| | read | 42:23 min | 3:27 min |
| | write_fail | 2:08 min | 25.66 secs |
| | write_hold | 27:08 min | 1:08 min |

**Abbildung9.** Ergebnisse: Adreß-Management-Einheit

Das vorgestellte Verfahren ist prinzipiell auf jede Art von Schaltung anwendbar, wie bereits erwähnt kann keine Verschlechterung der Verifikationsbedingungen erfolgen. In weiteren Untersuchungen stellte sich bei einer Reihe von realen Schaltungen heraus, daß die Methode eine signifikante Beschleunigung der Verifikationsaufgabe bewirkt. Es wurde eine bestimmte und wichtige Klasse von Schaltungen identifiziert, für die diese Art von Vorverarbeitung besonders gut geeignet scheint. Diese Klasse beinhaltet u.a. Speicher, FIFO's, Queues, Stacks, Bridges und Interface-Protokolle.

## 6 Stand der Technik

Unser Ansatz benutzt eine neue Technik zur Analyse von Datenabhängigkeiten und zur Word-Level Abstraktion. Den wissenschaftlichen Rahmen bildet eine formale Theorie von Bitvektoren fester Länge, die eine Erweiterung der Kerntheorie von Bitvektoren ist, die in [CMR96] und [Moe98] vorgestellt wird. Die Grundidee einer Granularitäts-Analyse findet sich ansonsten auch noch in [CMR97]. [Moe98] präsentiert eine Word-Level Entscheidungsprozedur für einzelne Gleichungen innerhalb der Kerntheorie mit Extraktions- und Konkatenationsoperator. Der Kern einer Minimale-Breiten-Abstraktion findet sich dort bei der Repräsentation bitweiser Boolescher Funktionen mittels Bitvektor-BDD's. Weitere Entscheidungsprozeduren für Bitvektor-Theorien finden sich weiterhin in [BP98] und [BDL98b], letztere behandelt zusätzlich Word-Level Addition. Kürzlich benutzte [ChH00] Word-Level ATPG und Modular

Arithmetic Constraint Solving Techniken innerhalb RT-Level Verifikationsverfahren. Letztlich benutzt z.B. [BV99b] einen gänzlich anderen Zugang zu Bit-Level Abstraktionen.

Der Autor dieses Verfahren ist sich keiner anderen Arbeit bewußt, die eine vergleichbare Granularitäts-Analyse für komplette RT-Level Schaltungsbeschreibungen oder eine ähnliche Minimale-Breiten-Abstraktion mit den hier geschilderten Eigenschaften benutzt. Auch ist ihm keine andere Arbeit bekannt, die vorher genannte Techniken oder ähnliche Bitvektor-Techniken nicht zur Verifikation, sondern zur Modell-Vorverarbeitung mit den hier genannten Merkmalen für die Verifikation anwendet.

## Literatur


[BDL98a] Clark W. Barrett, David L. Dill, Jeremy R. Levitt. Validity Checking for Combinations of Theories with Equality. *FMCAD'96*, pages 187–201. 1996.

[BDL98b] Clark W. Barrett, David L. Dill, Jeremy R. Levitt. A Decision Procedure for Bit-Vector Arithmetic. *DAC'98*, pages 522–527. 1998.

[BGV99] Randal E. Bryant, Steven M. German, Miroslav N. Velev. Exploiting Positive Equality in a Logic of Equality with Uninterpreted Functions. *CAV'99*, pages 470–482. 1999.

[BP98] Nikolaj Bjrner, Mark C. Pichora. Deciding Fixed and Non-fixed Size Bit-vectors. *TACAS'98*, pages 376–392. 1998.

[BV99a] Miroslav N. Velev, Randal E. Bryant. Exploiting Positive Equality and Partial Non-Consistency in the Formal Verification of Pipelined Microprocessors. *DAC'99*, pages 397–401. 1999.

[BV99b] Miroslav N. Velev, Randal E. Bryant. Bit-Level Abstraction in the Verfication of Pipelined Microprocessors by Correspondence Checking. *FMCAD'98*, pages 18–35. 1998.

[ChH00] Chung-Yang Huang, Kwang-Ting Cheng. Assertion checking by combined word-level ATPG and modular arithmetic constraint-solving techniques. *DAC'00*, pages 118–123. 2000.

[CMR96] David Cyrluk, M. Oliver Möller, Harald Rueß. An Efficient Decision Procedure for a Theory of Fixed-Sized Bitvectors with Composition and Extraction. *Ulmer Informatik-Berichte 96-08*, Fakultät für Informatik, Universität Ulm. 1996.

[CMR97] David Cyrluk, M. Oliver Möller, Harald Rueß. An Efficient Decision Procedure for the Theory of Fixed-Sized Bit-Vectors. *CAV'97*, pages 60–71. 1997

[Le98] Jeremy R. Levitt. Formal Verification Techniques for Digital Systems. *PhD Thesis at the Department of Electrical Engineering, Stanford University.* 1998.

[MdS97] João P. Marques Silva. Search Algorithms for Satisfiability Problems in Combinational Switching Circuits. *PhD Thesis, University of Michigan.* 1995.

[MdSS00] João P. Marques Silva, Karem A. Sakallah. Boolean Satisfiability Algorithms and Applications in Electronic Design Automation. *CAV'00, Invited Tutorial.* 2000.

[Moe98] M. Oliver Möller. A Decision Procedure for Hardware Verification. *Diploma Thesis, University of Ulm.* 1998.

[MR97] M. Oliver Möller, Harald Rueß. Solving Bit-Vector Equations. *FMCAD'98*, pages 36–48. 1998.

# A Anhang

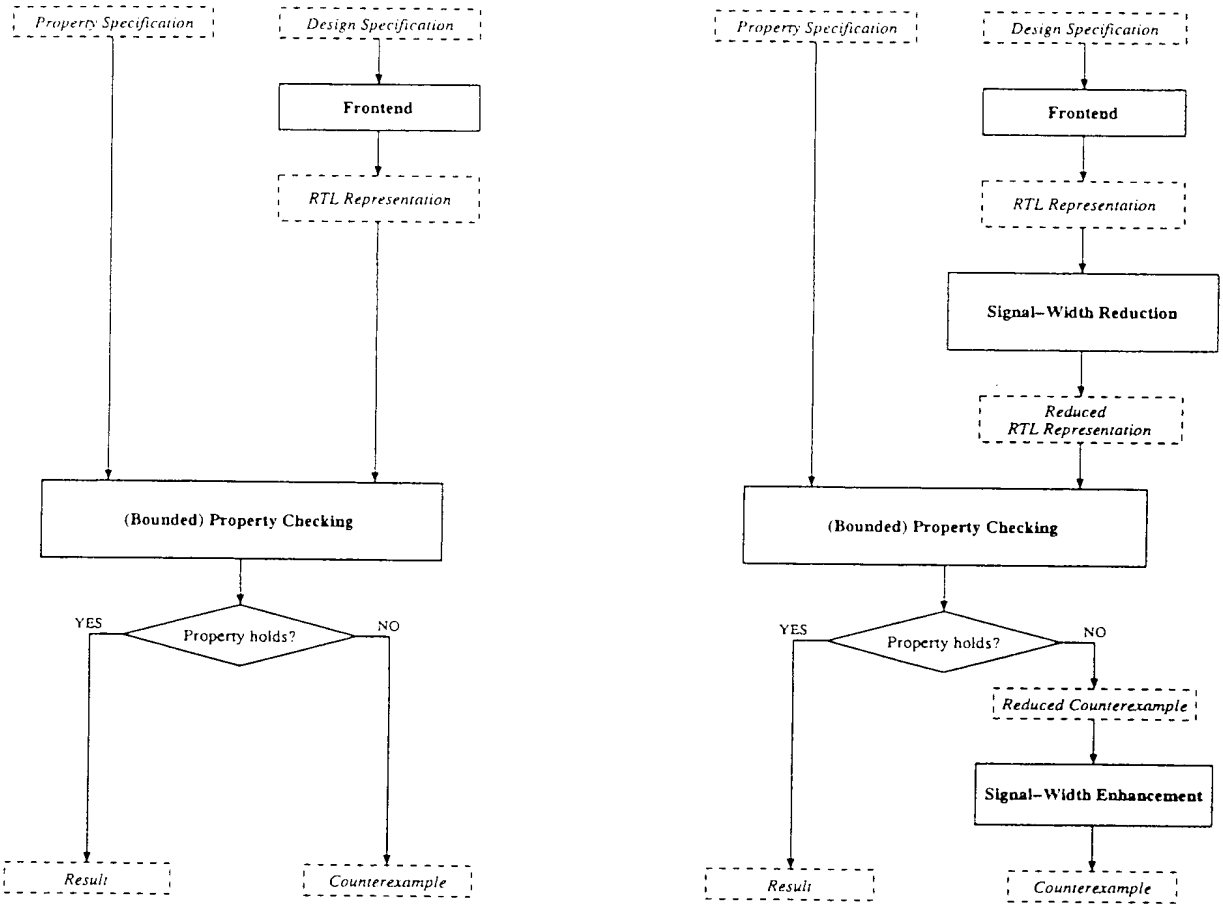
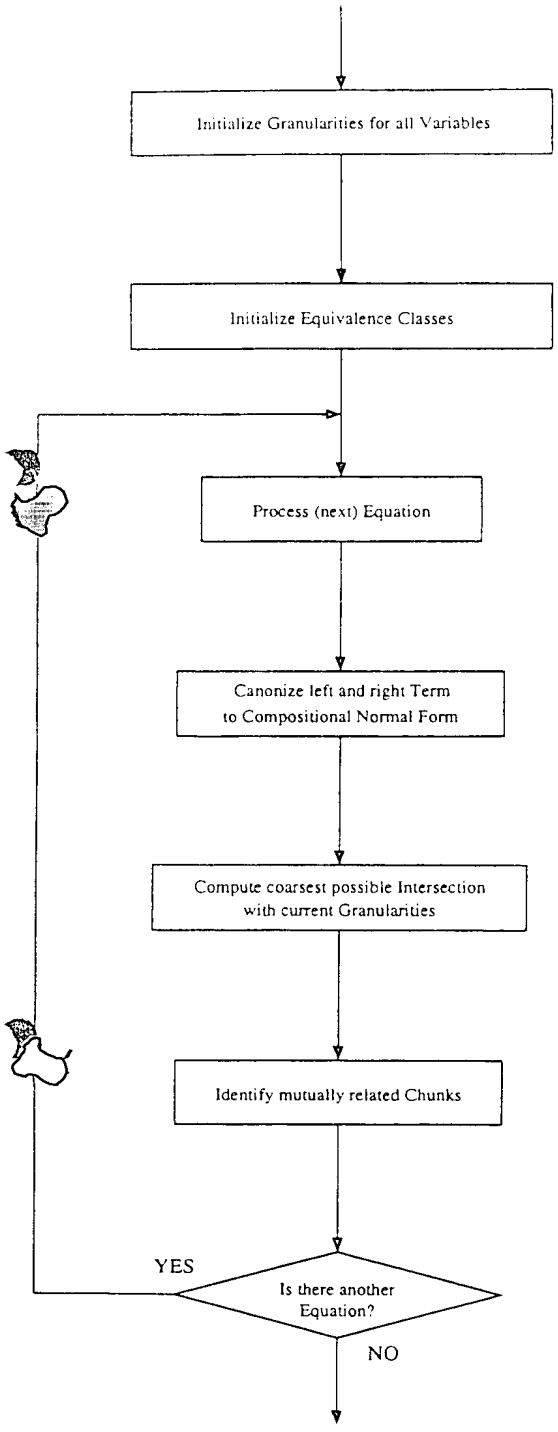



**Abbildung10.** Classical Property Checking (left) vs. Property Checking with Preprocessing (right)

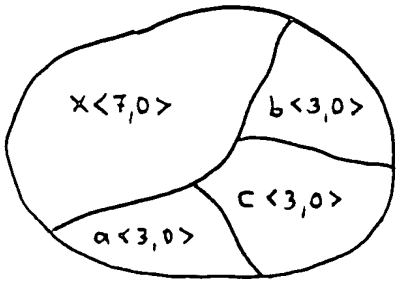


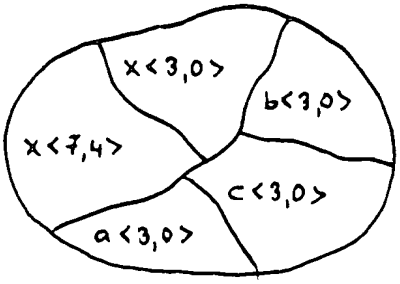

$x_{[8]}[7,2] \otimes x_{[8]}[1,0] \;=\; (a_{[4]} \otimes b_{[4]} \otimes c_{[4]})[11,4]$

$x_{[8]}[7,0] \;=\; a_{[4]}[3,0] \otimes b_{[4]}[3,0]$

$x_{[8]}[7,4] \otimes x_{[8]}[3,0] \;=\; a_{[4]}[3,0] \otimes b_{[4]}[3,0]$

$x_{[8]}[7,4] \longleftrightarrow a_{[4]}[3,0]$
$x_{[8]}[3,0] \longleftrightarrow b_{[4]}[3,0]$



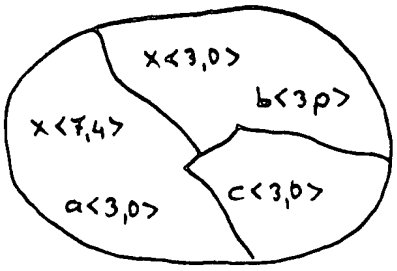
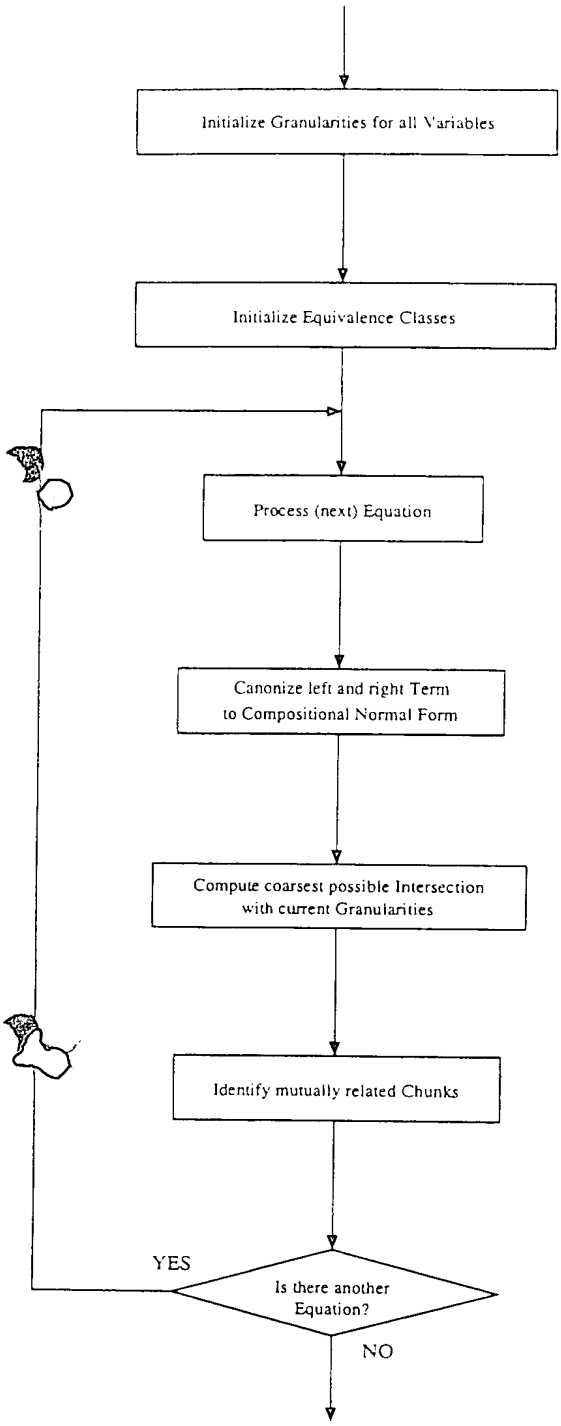




**Abbildung11.** Granularitätsanalyse (Flow + Beispiel 1)

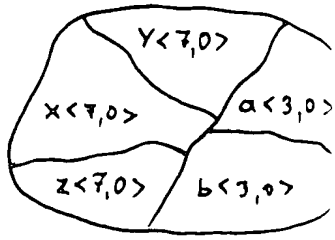


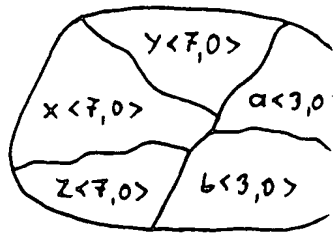

$$x_{[8]} = \text{ite}(a_{[4]} = b_{[4]},\ y_{[8]},\ z_{[8]})$$

$$x_{[8]} = \text{ite}(a_{[4]} = b_{[4]},\ y_{[8]},\ z_{[8]})$$

$$x_{[8]}[7,0] = \text{ite}(a_{[4]}[3,0] = b_{[4]}[3,0],\ y_{[8]}[7,0],\ z_{[8]}[7,0])$$

$$a_{[4]}[3,0] \longleftrightarrow b_{[4]}[3,0]$$
$$x_{[8]}[7,0] \longleftrightarrow y_{[8]}[7,0]$$
$$x_{[8]}[7,0] \longleftrightarrow z_{[8]}[7,0]$$



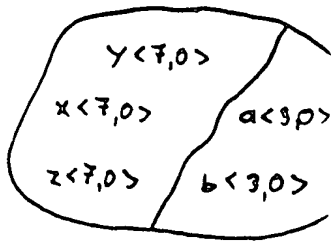
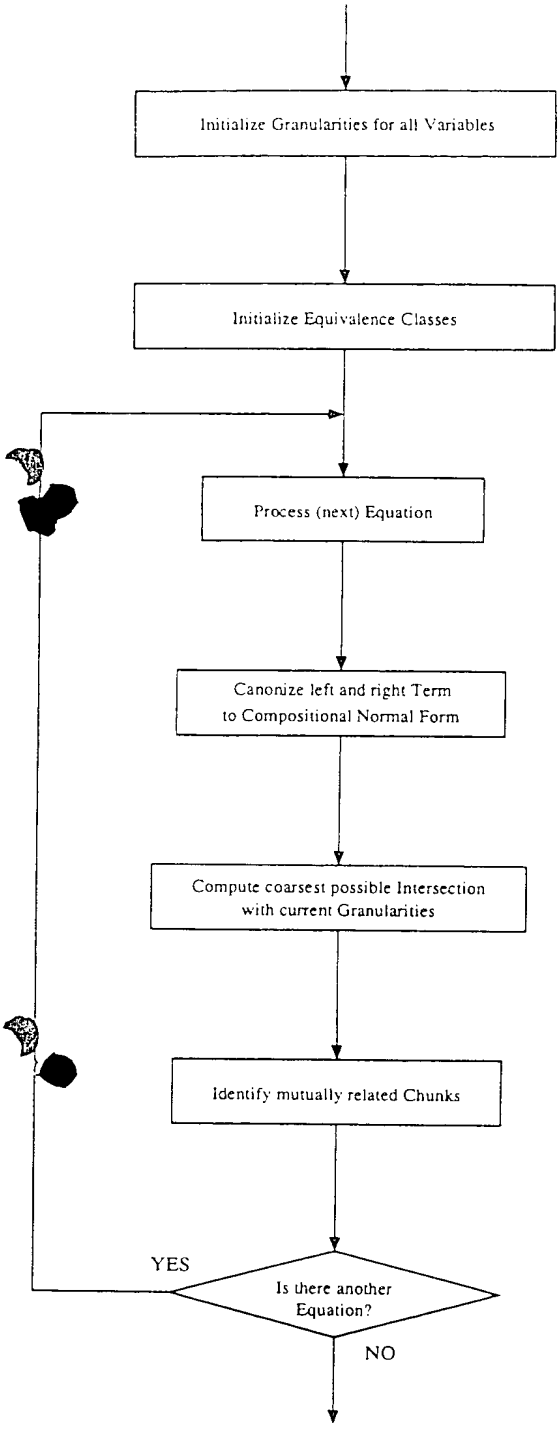




**Abbildung12.** Granularitätsanalyse (Flow + Beispiel 2)

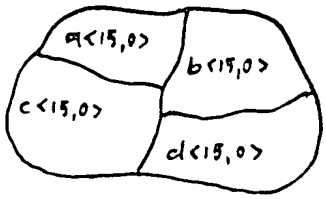


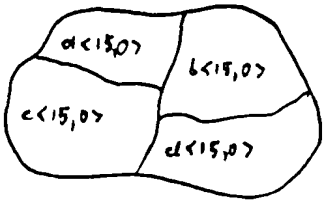

$a_{[16]}$ and $b_{[16]} = c_{[16]}$ or $d_{[16]}$

$a_{[16]}$ and $b_{[16]} = c_{[16]}$ or $d_{[16]}$

$a_{[16]}[15,0]$ and $b_{[16]}[15,0] = c_{[16]}[15,0]$ or $d_{[16]}[15,0]$

$a_{[16]}[15,0] \longleftrightarrow b_{[16]}[15,0]$
$a_{[16]}[15,0] \longleftrightarrow c_{[16]}[15,0]$
$a_{[16]}[15,0] \longleftrightarrow d_{[16]}[15,0]$



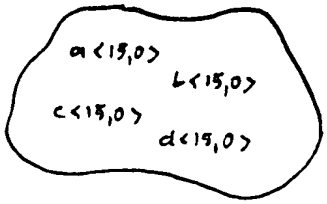
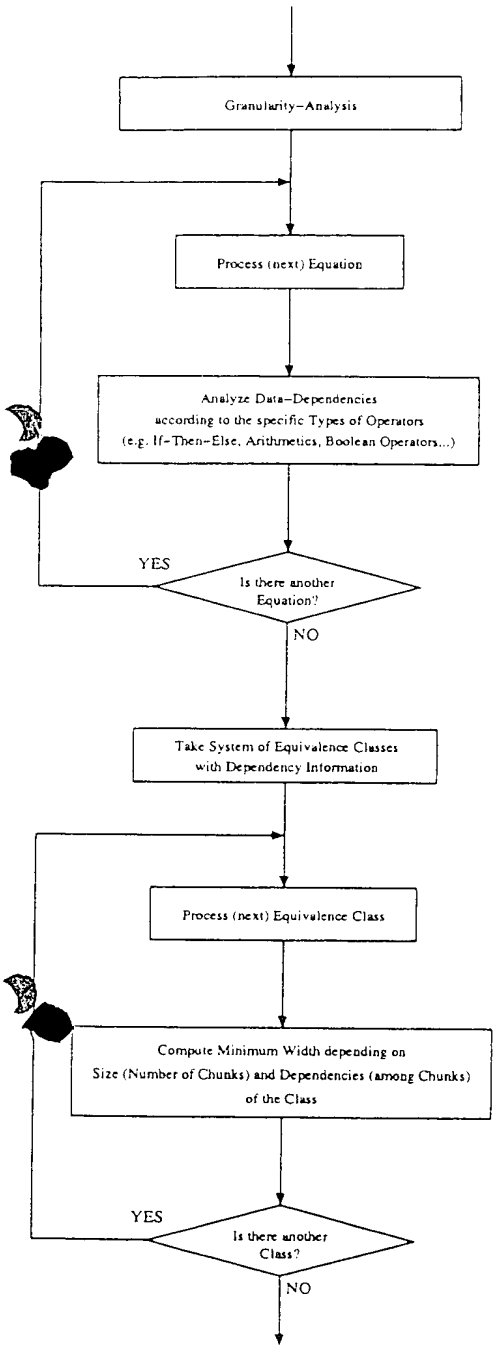




**Abbildung13.** Granularitätsanalyse (Flow + Beispiel 3)

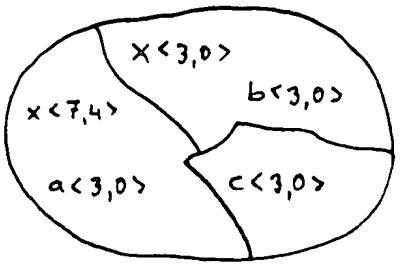


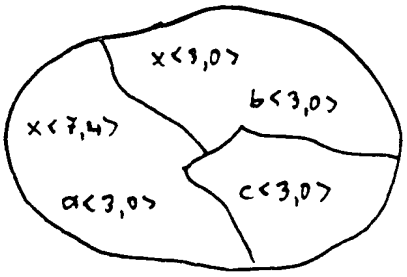

$x_{[8]}[7,4] \otimes x_{[8]}[3,0] \;=\; a_{[4]}[3,0] \otimes b_{[4]}[3,0]$

Keine operator-bedingten Abhängigkeiten

$C$, keine operator-bedingten Abhängigkeiten

$min \;=\; 1$



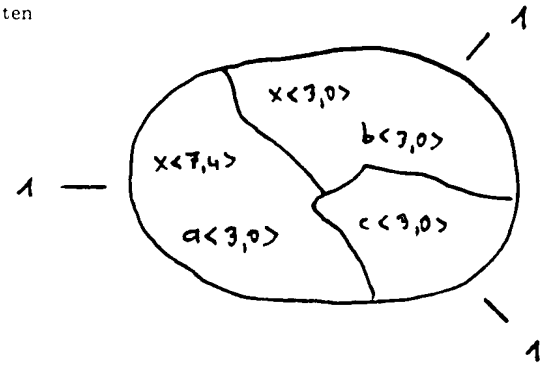
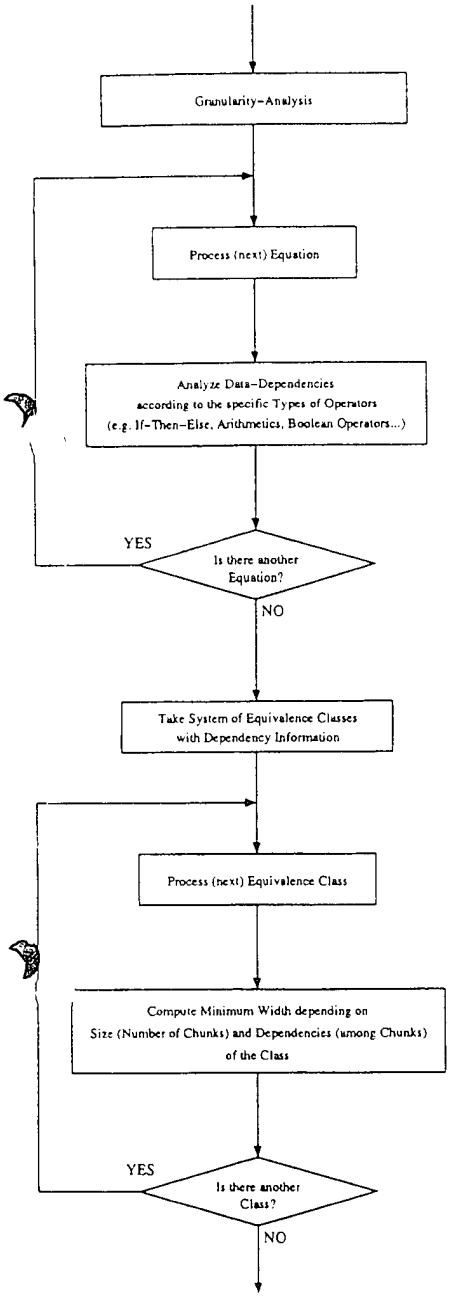




**Abbildung14.** Minimale-Breiten-Abstraktion (Flow + Beispiel 1)

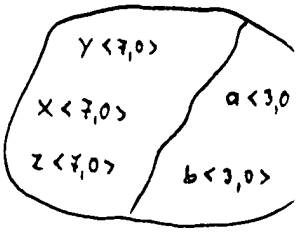


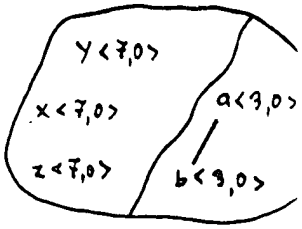

$x_{[8]} = \text{ite}(a_{[4]} = b_{[4]},\ y_{[8]},\ z_{[8]})$

Mögliche Ungleichung $a \neq b$

$C$, ungleichungs-bedingte Abhängigkeiten

$min = |C| - |\text{Zusammenhangskomponenten}|$



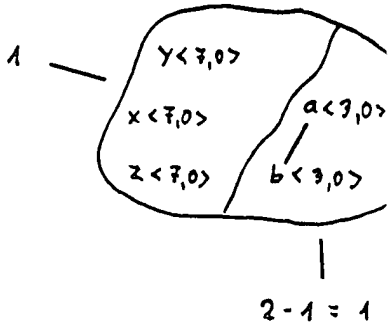
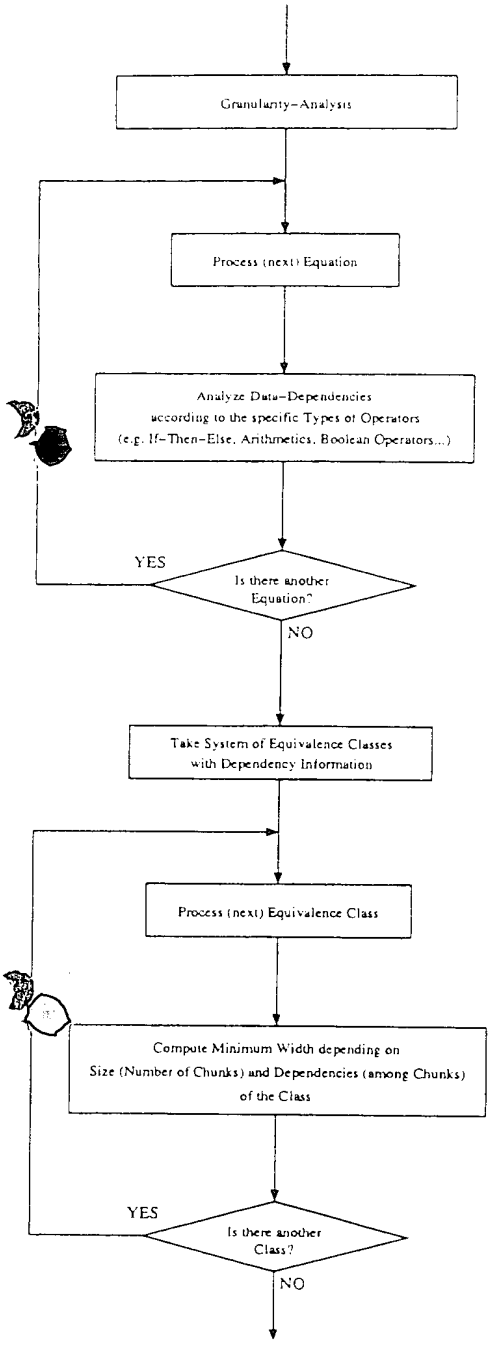




**Abbildung15.** Minimale-Breiten-Abstraktion (Flow + Beispiel 2)

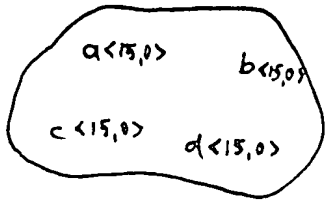


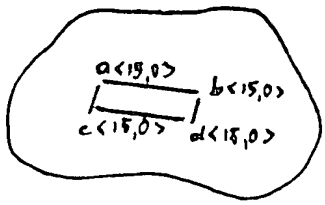

$a_{[16]}$ and $b_{[16]}$ = $c_{[16]}$ or $d_{[16]}$

Boolesch-bedingte Abhängigkeiten:
$B(a_{[16]}, b_{[16]}, c_{[16]}, d_{[16]})$

$C$, Boolesch-bedingte Abhängigkeiten

$min = \max_{B}(\min(B))$
$B$ Boolesche Zusammenhangskomponente von $C$



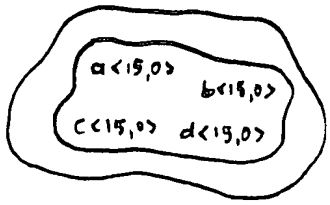
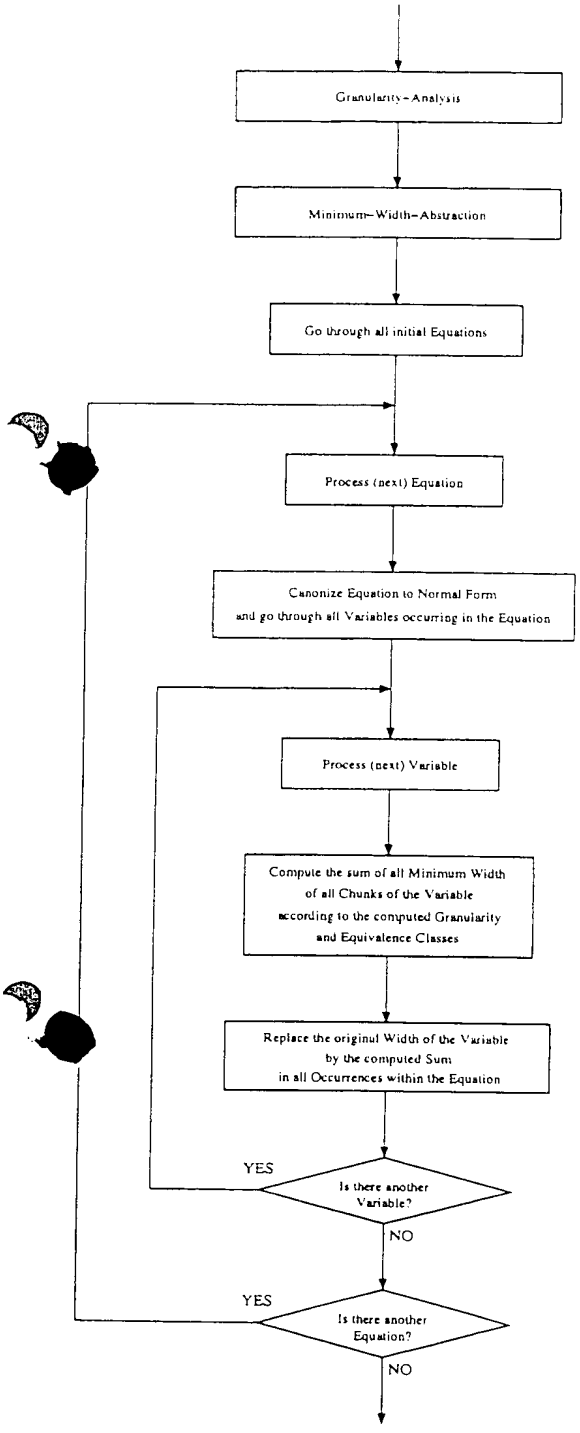
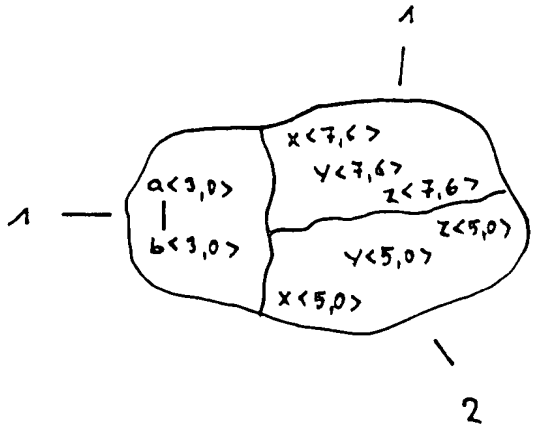




**Abbildung16.** Minimale-Breiten-Abstraktion (Flow + Beispiel 3)




$x_{[8]}\langle 7,6\rangle\langle 5,0\rangle$
$y_{[8]}\langle 7,6\rangle\langle 5,0\rangle$
$z_{[8]}\langle 7,6\rangle\langle 5,0\rangle$
$a_{[4]}\langle 3,0\rangle$
$b_{[4]}\langle 3,0\rangle$

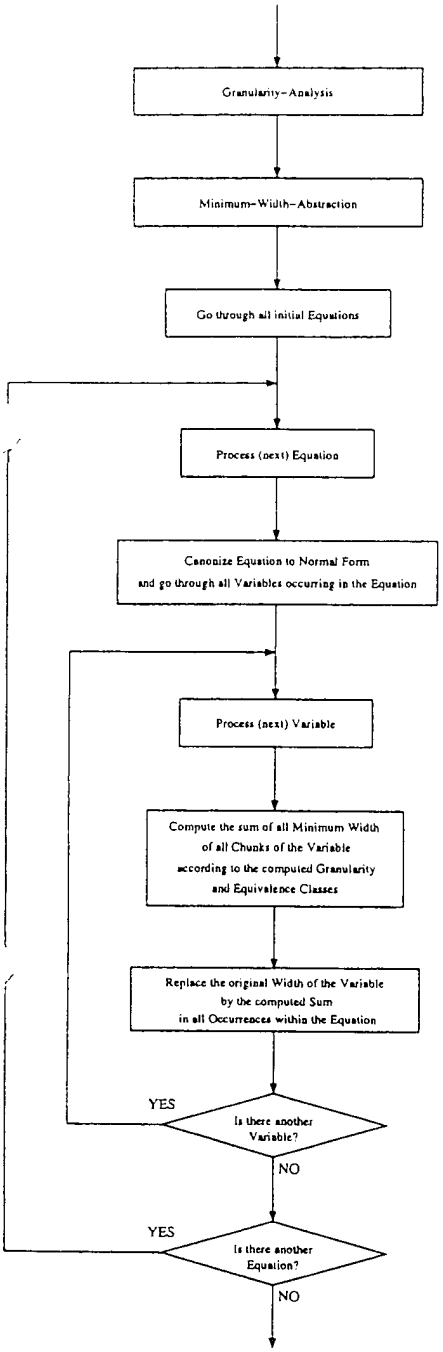



$x_{[8]} = \text{ite}(a_{[4]} = b_{[4]},\ y_{[8]},\ z_{[8]})$

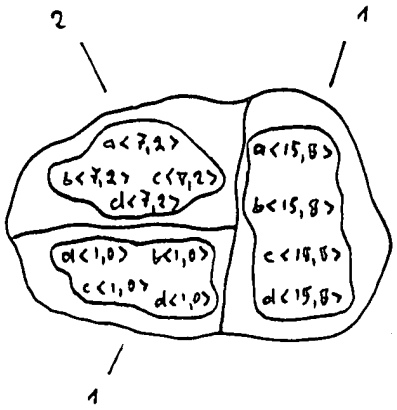
| $x_{[8]}$ | $y_{[8]}$ | $z_{[8]}$ | $a_{[4]}$ | $b_{[4]}$ |
|---|---|---|---|---|
| $\Sigma = 3$ | $\Sigma = 3$ | $\Sigma = 3$ | $\Sigma = 1$ | $\Sigma = 1$ |

$x_{[3]} = \text{ite}(a_{[1]} = b_{[1]},\ y_{[3]},\ z_{[3]})$

**Abbildung18.** Modell-Generierung (Flow + Beispiel 2)




$a_{[16]}\langle 15,8\rangle\langle 7,2\rangle\langle 1,0\rangle$
$b_{[16]}\langle 15,8\rangle\langle 7,2\rangle\langle 1,0\rangle$
$c_{[16]}\langle 15,8\rangle\langle 7,2\rangle\langle 1,0\rangle$
$d_{[16]}\langle 15,8\rangle\langle 7,2\rangle\langle 1,0\rangle$




$a_{[16]}$ and $b_{[16]} = c_{[16]}$ or $d_{[16]}$

$a_{[16]}$ $b_{[16]}$ $c_{[16]}$ $d_{[16]}$

$\Sigma = 4$ $\Sigma = 4$ $\Sigma = 4$ $\Sigma = 4$

$a_{[4]}$ and $b_{[4]} = c_{[4]}$ or $d_{[4]}$

**Abbildung19.** Modell-Generierung (Flow + Beispiel 3)

US 1003887006P1

Creation date: 12-03-2003
Indexing Officer: CTO - CUONG TO
Team: OIPEBackFileIndexing
Dossier: 10038870

Legal Date: 03-28-2003

| No. | Doccode | Number of pages |
|---|---|---|
| 1 | SRNT | 3 |

Total number of pages: 3

Remarks:

Order of re-scan issued on .....